INTRO TO NUMERICAL METHODS

Textbook (Python) Part I Chapter 1

What is a numerical method?

A math tool that calculates an approximation of a solution to a problem

- **Numerical analysis** is the study of algorithms that use ***numerical approximation*** (as opposed to symbolic manipulations) for problems of mathematical analysis.
- In numerical analysis, a **numerical method** is a mathematical tool designed to solve numerical problems, attempting at ***finding approximate solutions*** (to any level of accuracy) of problems rather than the exact ones.
- The ***implementation*** of a numerical method with an appropriate convergence check in a programming language is called a **numerical algorithm**.

https://matterofmath.com/tools/math-terms-definitions/#numerical-analysis; https://matterofmath.com/calculus/eulers-method/

Why numerical methods? Why an approximation?

- **Want** to solve for the equation: $y = f(t)$
  But it is difficult or impractical to find
  So, we mathematically approximate it

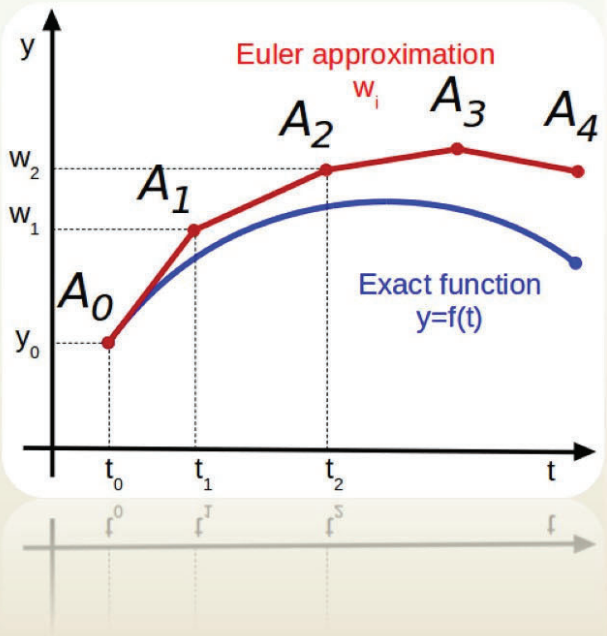
- **Easy:** $y' = \frac{dy}{dx} = x$

  Solve analytically: $y(x) =$

- **Real-world:** $\frac{dv}{dt} = g - \frac{c_d}{m}v^2$

  Solve analytically: $v(t) = \sqrt{\frac{gm}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}t\right)$

  Instead, how about we **approximate** this?



Why numerical methods? Why an approximation?

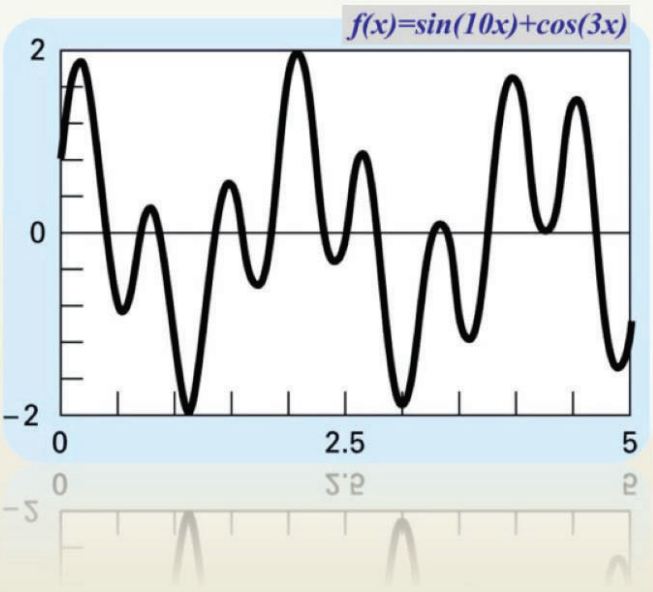
- **Want** to solve for the roots (find x)
  But it is difficult or impractical to find
  So, we mathematically approximate it

- **Easy:** $ax^2 + bx + c = 0$

  Solve analytically: $x =$

- **Not so easy:** $x = ?$
  $ax^5 + ax^4 + cx^3 + dx^2 + ex + f = 0$

- **How about this:** $\sin 10x + \cos 3x = 0$
  Instead, how about we **approximate** it?



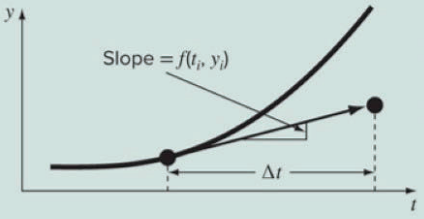
Topics

(Ordinary) differential equations

Given $\frac{dy}{dt} \approx \frac{\Delta y}{\Delta t} = f(t, y)$

solve for y as a function of t

$$y_{i+1} = y_i + f(t_i, y_i)\,\Delta t$$

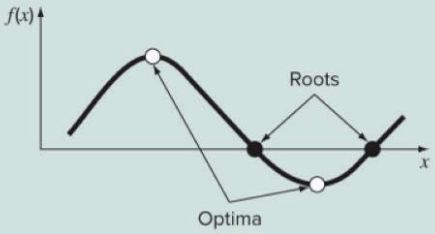


Roots and optimization

Roots: Solve for *x* so that *f*(*x*) = 0

Optimization:

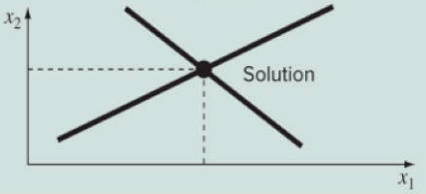
Solve for *x* so that *f*′(*x*) = 0



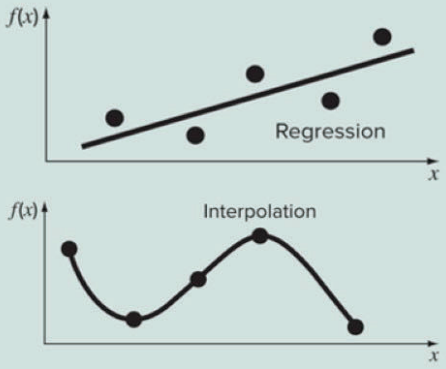
Linear algebraic equations

Given *a*'s and *b*'s, solve for *x*'s

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$



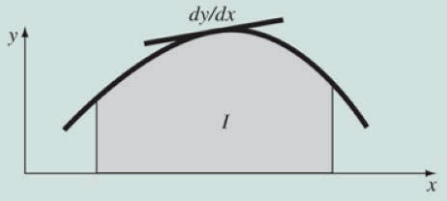
Curve fitting



Numerical integration & differentiation

Integration: Find the area under the curve

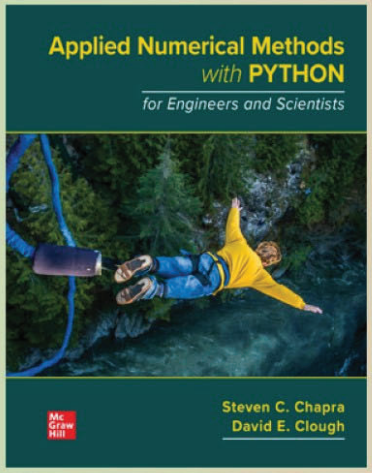
Differentiation: Find the slope of the curve

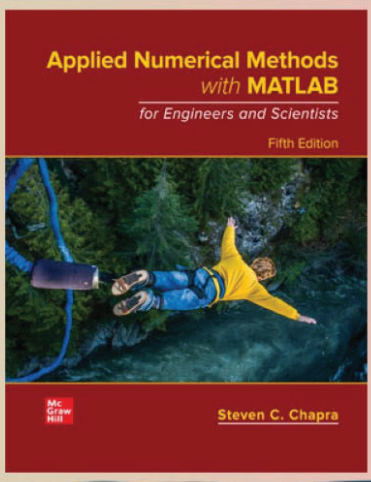


Required textbooks and software

Midterm

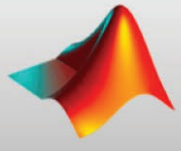
Aj.Pla





Final

Aj.Tee



MATLAB®

Evaluations and Assessments

WEEKS 1-8
Assignments and/or Quizzes
15%

MIDTERM
Exam FINEX
35%

WEEKS 9-15
Assignments and/or Quizzes
15%

FINAL
Exam FINEX
35%

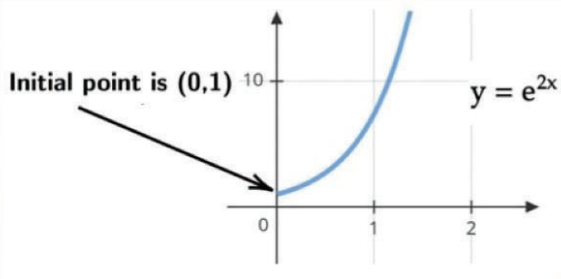
Problem: solve an ODE $dy/dx = x$

- **ODE: ordinary differential equation** is a differential equation with *only one* independent variable
- What does it mean to solve an ODE: finding an unknown curve whose equation satisfies the given ODE
  - *Given the derivative of a function f and an initial condition, find the function f*
- **Closed form solution** → solving it analytically using calculus $y(x) = \frac{x^2}{2} + C$
- **Numerical solution** → approximate it using Euler's method

Problem: solve an ODE $dy/dx = 2y$ with $y(0) = 1$

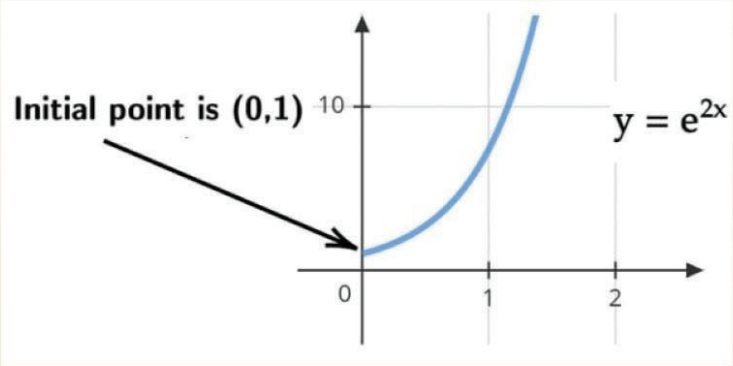
- **Closed form solution** → solving it analytically using calculus
- The analytical solution is $y = e^{2x}$
- Notice that $\frac{d}{dx}(e^{2x}) = 2e^{2x} = 2y$, which satisfies the given ODE

review TIME Using calculus can use solve this ODE?

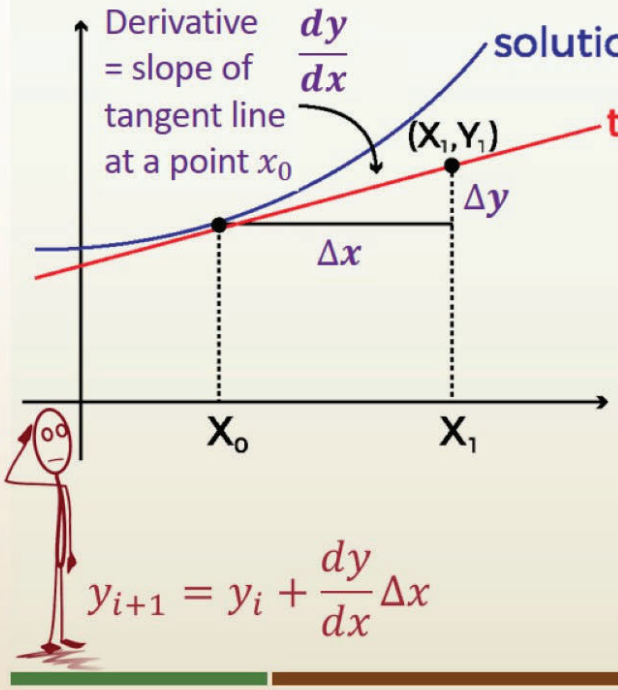


Problem: solve an ODE $dy/dx = 2y$ with $y(0) = 1$

- **Numerical solution** → approximating it e.g. using Euler's method
- To approximate a solution, build your own curve which roughly matches a solution → use the Euler's method $y_{i+1} = y_i + \frac{dy}{dx}\Delta x$ with $\Delta x = 1$



The Euler's method, estimating the next point



$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

- estimate the derivative $\frac{dy}{dx} \cong$
- change in $y = \Delta y =$
- next y = current y + Δy

  = current y +

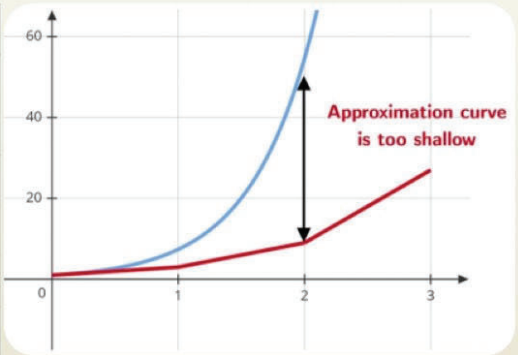
  = current y +

$$y_{i+1} = y_i + \frac{dy}{dx}\Delta x$$

Euler's method approximates ODE's

- **Problem:** *solve an ODE* $\frac{dy}{dx} = 2y$ with $y(0) = 1$
- To approximate a solution, build your own curve which roughly matches a solution → use the Euler's method $y_{i+1} = y_i + \frac{dy}{dx}\Delta x$ with $\Delta x = 1$

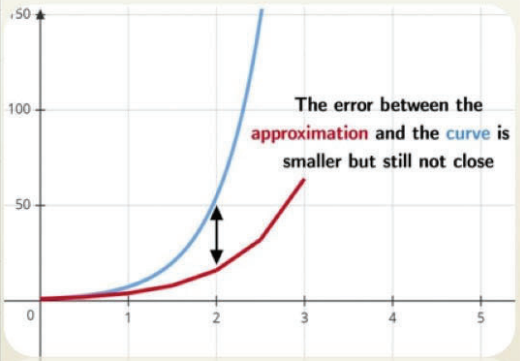
Iteration, i	x_i	y_i	y_i'
	$x_{i+1} = x_i + \Delta x$	$y_{i+1} = y_i + y_i'\Delta x$	$y_i' = 2y$
0	0	1	
1			
2			
3			
4			



- **Problem:** *solve an ODE* $\frac{dy}{dx} = 2y$ with $y(0) = 1$
- To approximate a solution, build your own curve which roughly matches a solution → use the Euler's method $y_{i+1} = y_i + \frac{dy}{dx}\Delta x$ with $\Delta x = 0.5$

Iteration, i	x_i	y_i	y_i'
	$x_{i+1} = x_i + \Delta x$	$y_{i+1} = y_i + y_i'\Delta x$	$y_i' = 2y$
0	0	1	
1			
2			
3			
4			
5			
6			
7			
8			

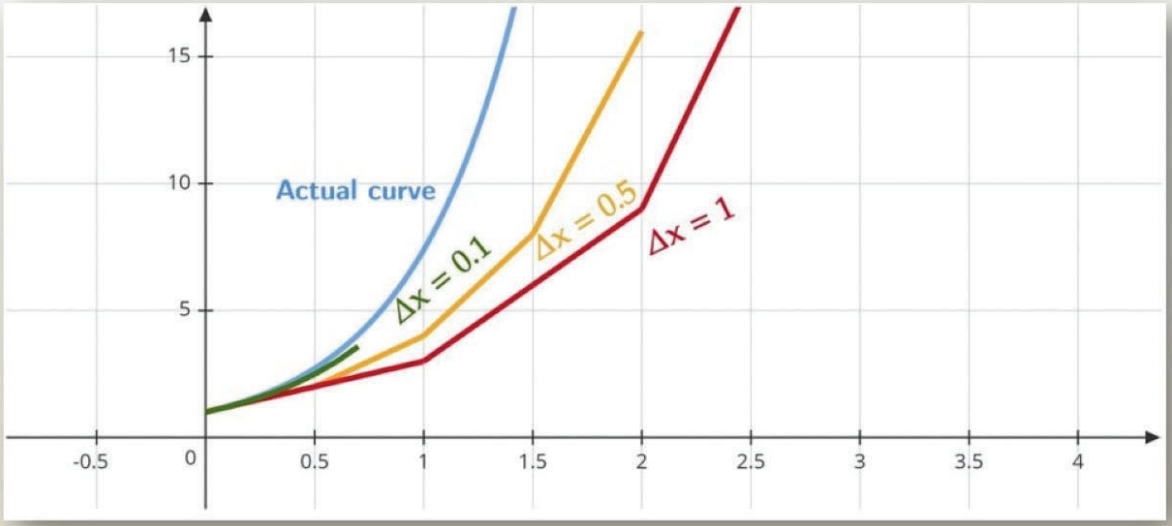
Approximate y(4)=?



https://matterofmath.com/calculus/eulers-method/

Euler's method approximates ODE's

- Use computers for most numerical applications → the number of *tiny*-step iterations needed to be accurate would take a *long* time to do by hand.



WORKED EXAMPLES

- Given $\frac{dy}{dx} = x^2y - 1$ and $y(0) = 1$, use Euler's method with step size 0.1 to estimate $y(0.1)$

- Given $\frac{dy}{dx} = x - y^2$ and $y(0) = 1$, use Euler's method with two equal steps to estimate $y(0.2)$

Hand-calculations vs. coding and programming

- **Hand-calculations**
  - Necessary for understanding how the algorithms work
  - Applicable for easy problems or for the first few steps

- **Coding and programming**
  - Needed for more complex (mathematic) problems
  - Needed for real-world problems (analytical solution is mostly unknown)

PRACTICE PROBLEMS HOMEWORK

- Approximate the solution to an ODE $\frac{dy}{dx} = e^{x-y} + 4x^3 e^{-y}$ at $x = 5.5$

  Use an initial condition $y(x_0 = -0.5) = 1$ and a step size of $\Delta x = 1.0$
- Repeat with a step size of $\Delta x = 0.5$
- Plot the graphs for both cases, then discuss their similarities and differences

Answer this question in My Courses

Your answer must be accurate to at least 5 decimal places

PRACTICE PROBLEMS REAL WORLD PROBLEM SOLVING

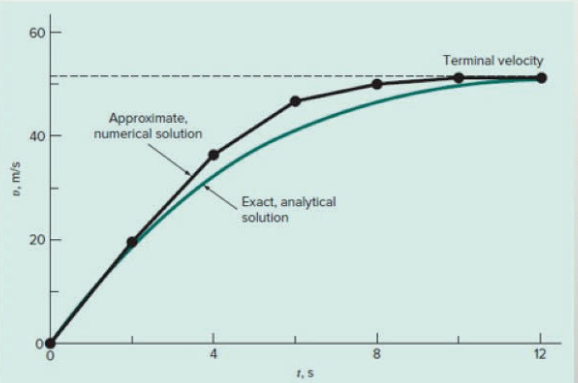
- Textbook Ex 1.1 – 1.2: **the bungee jumper problem** ODE $\frac{dv}{dt} = g - \frac{c_d}{m} v^2$

Analytical solution: $v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right)$ $\quad g = 9.81,\ c_d = 0.25,\ m = 68.1$

To do: approximate the solution using Euler's method with $v(t_0 = 0) = 0$ and $\Delta t = 2$

Fill in the table and plot the graphs to compare numerical & analytical solutions.

(expected resulting graphs are shown here, detail can be found in the textbook)



t	v
0	
2	
4	
⋮	
12	
∞	

Programming numerical methods with Python

- **Programming background** (from your last semester or earlier)
  - Variables, arrays, arithmetic operations, conditions, loops, and functions
  - Code will be provided: you need to be able to work with it and/or modify it
- **Programming environment** provided during the exam – we recommend that you have this set up on your machine and bring your notebook to class

- Alternative: this is convenient but will not be available during the exams!!

Google colab

01 Data types, numbers, assignments, operators, lists

```
# Integers
x = 10
y = -10

# Floats
z = 1.2345678
w = 3.
```

```
# Boolean
p = True
print(p or False)
                True
print(not p and p)
               False
```

```
# Arithmetic operations
print(x-y)          20
print(x+z)          11.2345678
print(y*z)          -12.345678
print(2**x)         1024
```

```
print(x/6)  1.6666666666666667
print(x//3) 3
print(x%3)  1
```

```
# Logical operations
print(x<=y)         False
print(x>z)          True
```

```
# List
L = [1, 3, 8, 9]
print(L[0])                  1
print(L[1:3]])           [3, 8]
print(L[-1])                 9

L[2] = 5
L.append(4)
print(L)       [1, 3, 5, 9, 4]
```

```
# Scientific notation
print(4.2e3)          4200.0
print(4.2e-3)         0.0042
```

02

Selection: sometimes you only want some lines of code to be run only if a condition is met, otherwise you want the computer to skip these lines.

```
if condition:
    statement
    statement
      ...
elif condition:
    statement
    statement
      ...
else:
    statement
    statement
      ...
following_statement
```

First condition: This is executed if the first condition is true

New condition: A new condition to test if previous condition isn't true

False branch: This is executed if none of the conditions are true

```
x, y = 5, 5
if x > y:
    print('x is greater')
elif x < y:
    print('y is greater')
else:
    print('x and y are equal')

# Prints x and y are equal
```


https://www.learnbyexample.org/python-if-else-elif-statement/

03

Iteration or loop: this is when a set of instructions is repeated a specified number of times (for-loop) or until a certain condition is met (while-loop).

```
for var in iterable:
    statement
    statement
      ...
else:
    statement
      ...
following_statement
```

Var: It takes items from iterable one by one

Iterable: It's a collection of objects (like a list, tuple etc.)

Indentation: Loop body must be indented

Loop body: It is executed once for each item in iterable

Else caluse: It is executed if the loop terminates naturally

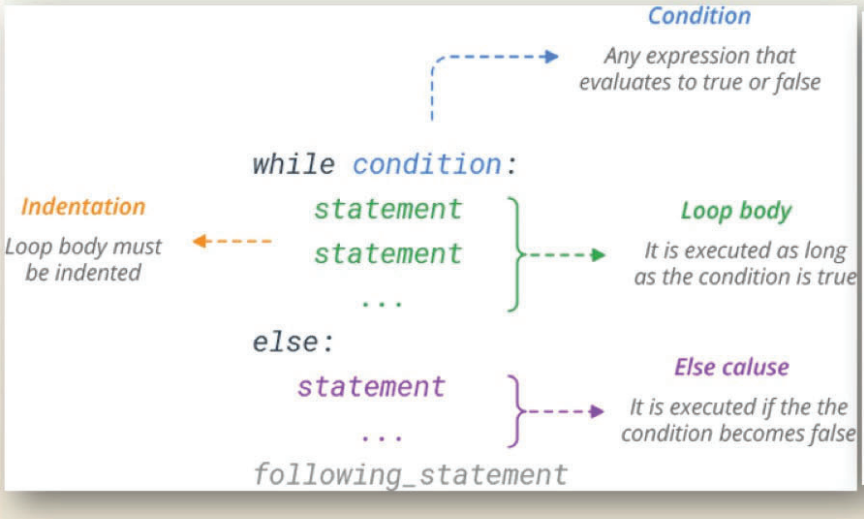
```
# Generate a sequence of numbers
for x in range(2, 7):
    print(x)
# Prints 2 3 4 5 6
```

```
# Increment the range with 2
for x in range(2, 7, 2):
    print(x)
# Prints 2 4 6
```


https://www.learnbyexample.org/python-while-loop/; https://www.learnbyexample.org/python-for-loop/;

03

Iteration or loop: this is when a set of instructions is repeated a specified number of times (for-loop) or until a certain condition is met (while-loop).



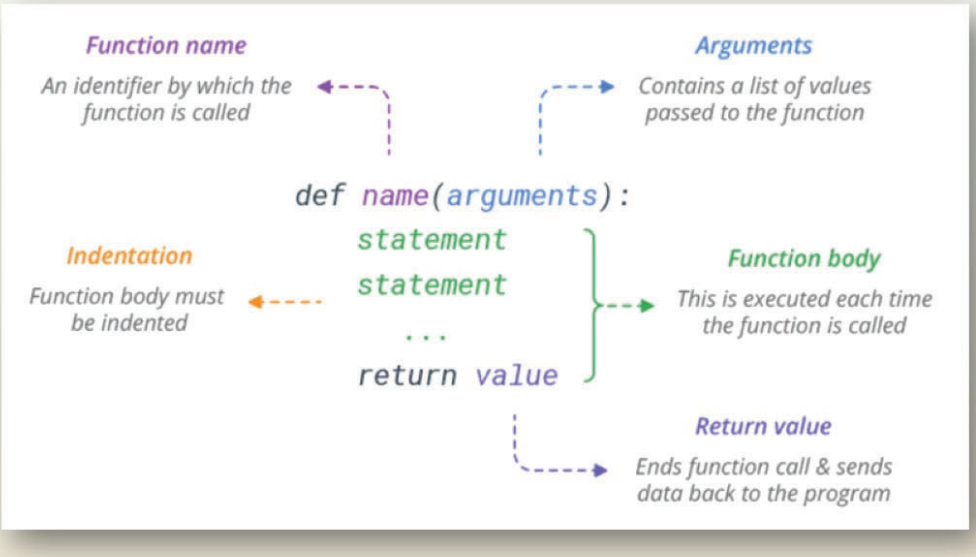

```
# Loop runs until the user enters 'stop'
while True:
    name = input('Enter name:')
    if name == 'stop': break
    print('Hello', name)

# Output:
# Enter name:Bob
# Hello Bob
# Enter name:Sam
# Hello Sam
# Enter name:stop
```


https://www.learnbyexample.org/python-while-loop/; https://www.learnbyexample.org/python-for-loop/;

04

Function: allow you to define a reusable block of code that can be used repeatedly in a program.




```
def pow(a,b):
  return a**b

print(pow(5,3))     125
```

```
def addsub(a,b):
  return a+b, a-b

add, sub = addsub(5,3)
print(add,sub)
                    8 2
```

```
f = lambda a,b: a/b
print(f(10,4))
                    2.5
```

Euler's method in Python

Iteration, i	x_i	y_i	y_i'
	$x_{i+1} = x_i + \Delta x$	$y_{i+1} = y_i + y_i' \Delta x$	$y_i' = 2y$
0	0	1	

- *Input*: initial x_0 and y_0, target value of x_f, and Δx
- *Process*: loop from x_0 to x_f with a step size of dx. Each iteration computes the next values of x and y
- *Output*: arrays of x and y

```
def euler(x0,xf,y0,dx):
  idx = 0
  x,y = [x0],[y0]
  while True:
    x.append(x[idx] + dx)
    y.append(y[idx] + deriv(x[idx], y[idx]) * dx)
    idx = idx+1
    if x[idx] >= xf-1e-9: break
  return x,y
```

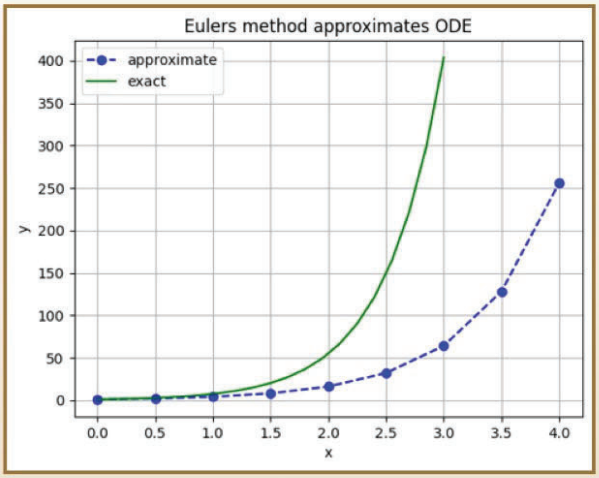
- Initialization, define ODE, then call the Euler function

```
deriv = lambda x,y: 2*y
x,y = euler(0,4,1,0.5)
```


Plotting graphs in Python

```
import numpy as np
import matplotlib.pyplot as plt
```

- *Define x-value:* an array of 21 evenly spaced points between x_i and x_f
- *Compute:* values of y based on the analytical solution of the ODE
- *Reminder:* in real-world, we may not have this analytical solution, we approximate it!




```
xx = np.linspace(0,3,21)
yy = np.exp(2*xx)
```

```
plt.plot(x,y, 'bo--', label='approximate')
plt.plot(xx,yy, 'g', label='exact')
plt.title('Euler''s method approximates ODE')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.legend(loc='upper left')
plt.show()
```